FlowX.AI

TECHNICAL PAPER

SEXTANT · A LABORATORY STUDY ON SYNTHETIC DOCUMENTS

Measuring Six-Nines Residual Error for LLM Decisions Behind Independent Deterministic Checks

No language model is accurate to six nines. We study a narrower quantity that can be measured: the residual error of one model step wrapped in deterministic checks, the probability that an output is wrong and accepted.

Bogdan Răduță
Head of Research, FlowX.AI
bogdan.raduta@flowx.ai

FlowX.AI · Applied Research · DOI: 10.5281/zenodo.24339368 November 2026 · Version 1.0

ABSTRACT

No language model is accurate to six nines, yet process automation is often described with such figures. We study a narrower quantity that can be measured: the residual error of one model step wrapped in deterministic checks, defined as the probability that an output is wrong *and* accepted. On synthetic payment documents with truth known by construction, one model (Claude Haiku 5.5) was wrong on about 9% of documents. We report four results.

First, the cheap way to certify a low residual, multiplying the model's error rate by the checks' miss rate on injected errors, failed: it bounded the residual at 9.19×10^{-4} while a direct count on 300,000 documents found 4.91×10^{-3}, 5.3 times higher. The model's errors were not like injected errors. Every wrong output that passed the checks was of one kind, located exactly where the checks could not see.

Second, one further deterministic rule closed that gap, and a direct zero-failure test on fresh documents found no wrong accepted output in 300,000 and then in 3,000,000 documents, a one-sided 95% upper bound of 9.99×10^{-7} per output processed for the 2,000 vendors of the test world. Because errors cluster by vendor, that bound does not carry over to new vendors (about 2×10^{-5} for a new set). A repeat with the same model through the provider's interface, on a fresh world of 50,000 vendors with about 60 documents each again found no wrong accepted output in 3,000,000 documents (bound 9.99×10^{-7}), and 1.78×10^{-6} for a new set of vendors from the same generator. The price was automation: the share of outputs accepted without review fell from 91.6% to 69.7% on the 2,000-vendor world.

Third, the same checks ported to a FlowX.AI integration workflow, run on the FlowX.AI Platform in a demo (non-production) environment with a second model (GPT-6 Luna), gave verdicts identical to the reference implementation on every one of 2,999,805 documents run end to end, and accepted 0 wrong outputs among them, a one-sided 95% bound of 9.99×10^{-7} per output processed on the 2,000-vendor world. A second pre-registered platform count, on the 50,000-vendor world, accepted no wrong output in 599,952 documents, a bound of 5.58×10^{-6} for a new set of vendors: five nines, not six. Outputs the checks could not verify were not accepted; the use case decides whether such an output is rejected, retried or sent to a person. They were the model's errors (about 8%) and correct outputs the ledger could not tell apart from another invoice (about 22% on this deliberately hard synthetic mix). Fourth, on 387 real invoices most of the model's errors (69%) were of a kind injected errors never produce.

The bounds below 10^{-4} concern one synthetic task. They support no statement about production systems or real invoices, and the real-invoice results are percent-level. What they do support is a method and a warning: a residual error rate is a property of a model, a prompt, a set of checks and a document mix together, and it has to be counted, not composed.

KEYWORDS residual error · deterministic checks · zero-failure testing · injected errors · document extraction · language models

SEXTANT v1.0 · November 2026 · DOI: 10.5281/zenodo.24339368

1 Introduction

Enterprise automation built on language models is frequently sold on accuracy figures such as 99.9999%. No model reaches that figure on its own. A system can, in principle, if the model's output is accepted only after deterministic checks against systems of record, and everything else is rejected, retried or sent to a person. RAILS[23] states the design rule: a generative step "must be followed by deterministic validation or extraction", and it shows that the *path* a case takes can be fixed by construction even when individual nodes are not deterministic. It also notes what structure alone does not give: "the schema guarantees a valid label, not a consistent judgment". RAILS measures validity and stability, not correctness. This paper takes the next step and asks how much error is left after the checks, and how one would know.

> *A residual error rate is a property of a model, a prompt, a set of checks and a document mix together, and it has to be counted, not composed.*
>
> SEXTANT · CORE THESIS

The quantity of interest is the residual error $r = P(\text{wrong and accepted})$ per output processed. Demonstrating $r \le 10^{-6}$ by direct testing needs 2,995,731 outputs with no failure at 95% confidence, and 10^{-5} needs 299,572[6]. That is expensive when truth has to come from people. Figure 1 shows the system. A tempting shortcut is to write $r = \varepsilon m$, measure the model error rate ε on a modest labelled sample, and measure the checks' miss rate m on cheap *injected* errors. The shortcut is valid only if injected errors are caught as often as the model's own errors. That assumption is the subject of this study.

FIG. 1 · THE SYSTEM UNDER TEST

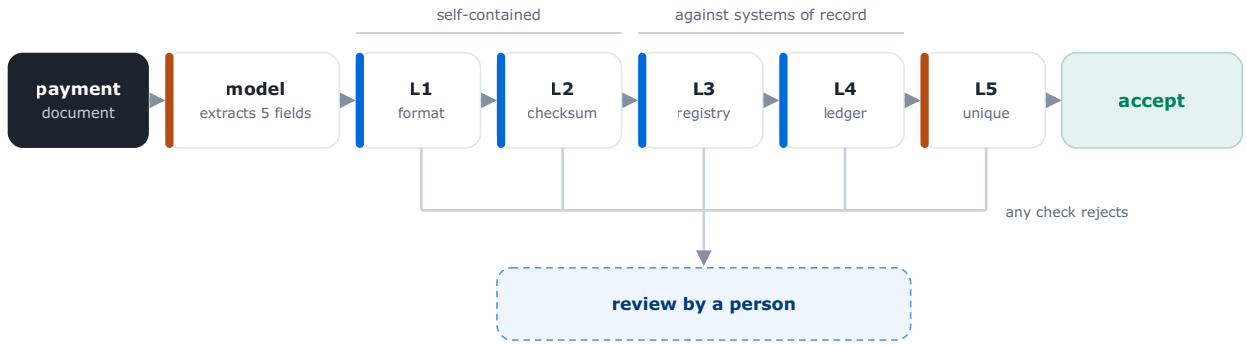

Figure 1. The system under test. One model step, then deterministic checks. An output is accepted only if every check accepts. In this study anything rejected is assumed to go to a person; a deployment may instead reject it outright or retry it. L5 was added after the first run (Section 7). The residual error r is the share of outputs that are wrong and still reach "accept".

1.1 Contributions

(1) Definitions and a composed bound with its two assumptions made explicit (Section 3).

(2) A simulation showing that the bound holds when its assumptions hold and fails under dependence and drift (Section 4).

(3) A pre-registered laboratory test on 6,600,000 synthetic documents in which the injected-error assumption failed, a blind spot was found by direct counting, and one rule closed it; the closed result was then repeated on a world of 50,000 vendors to bound the residual for vendors not seen before (Sections 5 to 7).

(4) A direct count of 3,000,000 documents run end to end through the same checks as a workflow on the FlowX.AI Platform, with a second model, in which the platform's verdicts equalled the reference implementation's on every finished run and no wrong output was accepted, and a second count of 600,000 documents on the 50,000-vendor world that bounds the platform's residual for vendors not seen before (Section 8).

1.2 Scope

Everything here is a laboratory result on synthetic documents, with one prompt and mainly two models: Claude Haiku 5.5 through the provider's interface for the laboratory runs, and GPT-6 Luna inside the platform workflow. The platform runs used a demo (non-production) environment, not a production system. Section 11 lists what cannot be concluded.

2 Related work

Ultra-high reliability

Demonstrating very low failure rates by testing alone is infeasible for many systems[2], and decomposition helps only if independence between components can itself be shown[16]. SEXTANT does not escape this argument. It tests the decomposition directly and finds it wanting on this task. Zero-failure testing has been applied to machine-learning classifiers at failure rates of 0.2% and above with at most a few thousand samples[7]. The largest error-free language-model run we found is a puzzle of about a million dependent steps solved by voting[19]; it reports no confidence bound and has no abstention. We found no prior direct count of a residual at 10^{-5} or below for a step that contains a language model.

Correlated failures

Redundant versions fail together more often than independence predicts[4, 13], and this holds for versions written by coding agents[22]. Kim et al.[11] report that language models agree on about 60% of the cases where both are wrong. Per-stage risk guarantees for agent pipelines do not compose automatically[15]; that result is at a 10% risk level, ours at 10^{-3} to 10^{-6}.

Injected and natural errors

Seeded faults are easier to find than real ones[12], many injected faults do not represent residual faults[20], and mutants correlate with real faults but leave a share uncoupled[8, 10]. For language models, synthetic errors differ from a model's own errors enough to defeat training on them[25]. Our hypothesis H4 is the analogue for deterministic checks on model outputs; what is new here is the measured miss-rate ratio and the comparison of a composed bound with a direct count.

Risk control with abstention

Conformal and selective risk control give guarantees for structured model outputs when abstention is allowed, and prove that a share of abstentions is unavoidable when the base error exceeds the target[14]. In document extraction, per-field selective risk control on real receipts[5] and calibrated straight-through processing of financial documents[9] work at risk levels of 5 to 20% and use learned confidence scores; in the latter, format checks such as the IBAN checksum are inputs to the score rather than gates. Like our bounds, these assume exchangeability between calibration and use. SEXTANT differs in gating on checks against systems of record and in counting the residual directly at levels four to five orders of magnitude lower, which is possible only because truth is known by construction.

Accuracy claims

Legal research tools marketed as hallucination-free were found to give incorrect or misgrounded answers in 17 to 33% of queries[17], and error rates estimated with a language model as judge can be badly miscalibrated[18]. We use no model as a judge.

Systems

RAILS[23] fixes the execution graph and classifies node types on a determinism gradient (deterministic, bounded, generative); extraction is a bounded node. HALO[21] describes layered oversight; both are earlier work by the present author and colleagues. Formal verification of structured queries is

offered commercially[1]. SEXTANT is complementary to these: it measures what such designs leave behind.

3 Estimands and the composed bound

For one output: $\varepsilon = P(\text{wrong})$; $m = P(\text{all checks accept} \mid \text{wrong})$; coverage $c = P(\text{accepted})$; residual $r = P(\text{wrong and accepted}) = \varepsilon m$; and $r_{acc} = r/c$, the error rate among accepted outputs, which is what a customer experiences. A low r can be bought by rejecting almost everything, so c is always reported with r.

Direct bound

Count wrong accepted outputs among n and take the one-sided 95% Clopper-Pearson upper bound[3]. With zero failures this is about $3/n$[6] (Figure 2).

FIG. 2 · WHAT A COUNT BUYS

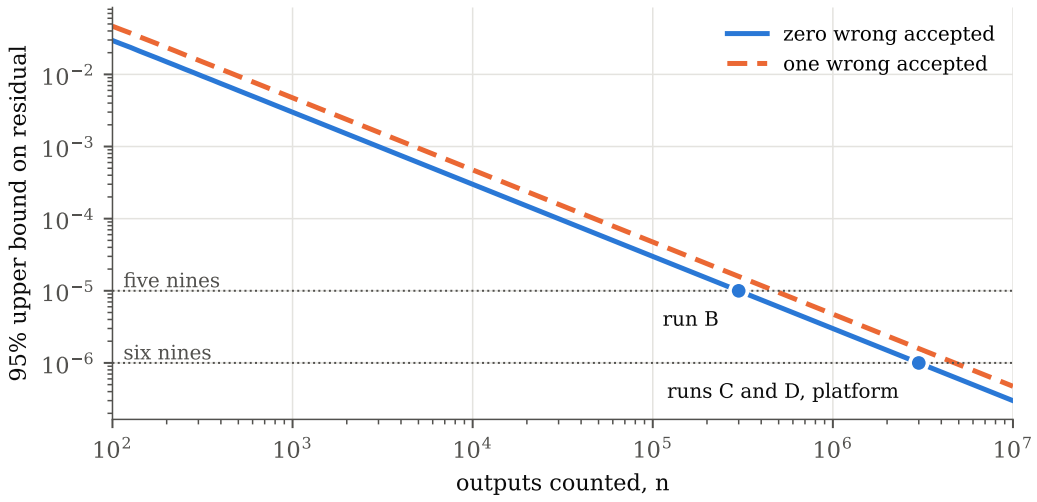

Figure 2. What a count buys. The one-sided 95% upper bound on the residual against the number of outputs counted, with zero and with one wrong accepted output. Five nines needs about 300,000 clean outputs; six nines about three million, and one failure raises that to about 4.7 million. Points are Run B and, at one place because each counted about three million outputs with zero wrong accepted, Runs C and D and the platform count (Section 8).

Composed bound

$U_r = U_\varepsilon(\alpha/2)\, U_m(\alpha/2)$, where U_m is measured on injected errors. It needs (A1) that injected errors are missed as often as real ones, and (A2) that the joint miss rate of the layers is measured jointly and not as a product of per-layer rates.

4 Simulation

With known ground truth, over 243 exchangeable configurations the composed bound covered the true residual in at least 0.984 of replications (mean 0.997). This follows from the union bound and checks the implementation. Two failures matter in practice. Under drift the certificate does not hold: when the error rate in use is twice the calibration rate, 59% of configurations fall below nominal coverage, and 79% at five times. Under dependent layers, a bound built on the product of per-layer miss rates covered the truth in none of the replications in every feasible configuration, while the bound on the measured joint miss rate covered it in at least 0.988; the true joint miss rate was 6.7 to 80 times the product. With two readers that miss together 60% of the time, the factor is 12.

5 Laboratory setup

Task

From one supplier payment document, return the payee, IBAN, amount, currency and invoice reference as JSON. An output is wrong if the IBAN, amount, currency or reference differs from the truth after normalisation.

Synthetic world

A fixed seed generates 2,000 vendors, each with one IBAN and one to six open orders. For 30% of vendors all orders carry the same amount. Each document is rendered for one order, with distractors at fixed probabilities (the payer's IBAN, another open invoice of the same vendor, a previous balance, an old IBAN marked as not to be used), at four difficulty levels in equal shares: clean; varied formats and three languages; simulated character-recognition noise on 3% of digits; and noise on 8% with sections in random order. Truth is known by construction. There are no human labels and no model acting as a judge.

Model

Claude Haiku 5.5, one fixed system prompt, low effort, through the provider's batch interface. A refusal, a cut-off or unparsable output counts as wrong.

Checks

All deterministic. L1 format; L2 IBAN checksum (ISO 7064, mod 97); L3 vendor registry (the IBAN is registered and the payee matches the registered name); L4 order ledger (the reference is an open order of that vendor with the same amount and currency). An output is accepted when all layers accept.

Order of work

Each phase's plan, criterion and analysis script were committed to the study's git repository before its first model call; the commit history is the record. L5 was designed after Run A had been analysed, from Run A's wrong accepted outputs, and committed together with the Run B plan before any Run B call. Runs B and C use documents that no earlier run had seen. Analyses marked exploratory were added after the data they use.

Protocol

Plans, analysis code and pass criteria were committed before each run. Each document was processed once. The three runs used disjoint document ranges. Total model cost was $504. Run D (Section 7) used a second world built the same way with a different seed and cost $423.

6 Result 1: injected errors understated the residual

On 300,000 documents the model was wrong on 9.0% (from 2.9% and 2.6% on the two clean levels to 11.0% and 19.5% on the two noisy ones). The four checks accepted 91.5% of outputs.

For each document we also made one injected error by applying a mutation operator, drawn uniformly from thirteen operators fixed in advance, to the true record. The checks missed 0.97% of injected errors and 5.45% of the model's errors: a ratio of 5.60 (95% CI 5.27 to 5.96). The pre-registered criterion required this ratio to lie within [0.5, 2]; assumption A1 was rejected. The composed bound was 9.19×10^{-4}. The direct count was 1,473 wrong accepted outputs, a rate of 4.91×10^{-3}, which is 5.3 times the bound (Figure 3).

FIG. 3 · RUN A: INJECTED ERRORS AGAINST THE MODEL'S OWN

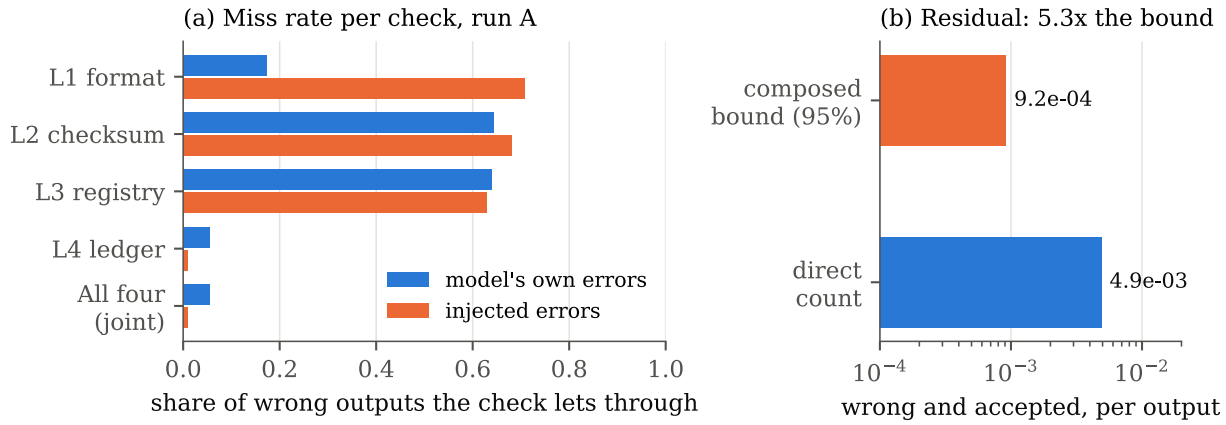

Figure 3. Run A. (a) Share of wrong outputs each check lets through, for the model's own errors and for injected errors. Format and the ledger miss the two kinds at very different rates. (b) The composed bound against the direct count of wrong accepted outputs.

Where the errors were

All 1,473 wrong accepted outputs were of one kind: the model returned the reference of *another open invoice of the same vendor that had the same amount*. That record passes every check, because the IBAN, the payee, the amount and the currency are right and the reference is a real open order. The model made this confusion in 4.2% of the documents where such an invoice was printed with the same amount, and in none of the 88,659 documents where the other invoice had a different amount. An injected error of the same class picks any other reference in the document and is caught about 70% of the time. The model's version is never caught. The model errs where the document is ambiguous, and that is where the ledger is ambiguous too.

Calibrating the mutations did not help

As a declared exploratory analysis, we reweighted the injected errors to the class mix seen in half of the model's errors and compared with the other half. The ratio was still 3.13 (bootstrap CI 2.80 to 3.47). Matching the mix of error classes is not enough when, within a class, the model selects the instances the checks cannot see.

7 Result 2: one more rule, tested directly

We added a fifth deterministic layer, L5: reject when the vendor that owns the IBAN has more than one open order with the record's amount and currency. The rule uses the ledger only. It was specified from the first run and then tested on fresh documents, with the criterion fixed in advance.

Table 1. Three runs on disjoint synthetic documents. Run A figures for five checks are in brackets because L5 was designed on that run; they are not a test.

	Run A	Run B	Run C
Documents	300,000	300,000	3,000,000
Model wrong, ε	9.0%	8.9%	8.8%
Four checks: accepted	91.5%	91.5%	91.6%
Four checks: wrong accepted	1,473	1,179	11,996
Five checks: accepted	(69.5%)	69.6%	69.7%
Five checks: wrong accepted	(0)	0	0
Upper bound on r, 95%	–	9.99×10^{-6}	9.99×10^{-7}
Upper bound on r_{acc}, 95%	–	1.77×10^{-5}	1.76×10^{-6}

With five checks there was no wrong accepted output in 300,000 fresh documents, and then none in 3,000,000 more. The second count gives $r \le 9.99\times10^{-7}$ per output processed at 95% confidence, which is below 10^{-6}. Among accepted outputs the bound is 1.76×10^{-6}, which is above 10^{-6}: the six-nines statement holds per output processed and not yet per accepted output. In Run C the four checks alone let 11,996 errors through (4.00×10^{-3}), so the fifth layer was doing the work (Figure 4). Results were stable across 10 blocks of 300,000 documents (ε from 8.7% to 9.0%; 1,141 to 1,261 four-check escapes per block; zero five-check escapes in each; Figure 5).

Control arm: the model alone (exploratory, after the data)

The same Run C outputs, with checks added one at a time (Table 2). Accepting every output, the model alone let through 265,048 wrong results, 8.8%. Each layer removes some; four layers leave 0.40%; the fifth removes the rest at a cost of 22 points of coverage.

Table 2. Run C (3,000,000 documents): the same model outputs under cumulative acceptance rules.

Acceptance rule	Coverage	Wrong accepted	Residual
Model alone	100.0%	265,048	8.8×10^{-2}
+ L1 format	92.6%	43,006	1.4×10^{-2}
+ L2 checksum	91.6%	13,824	4.6×10^{-3}
+ L3 registry	91.6%	13,455	4.5×10^{-3}
+ L4 ledger	91.6%	11,996	4.0×10^{-3}
+ L5 uniqueness	69.7%	0	$\leq 1.0\times10^{-6}$

Can the model screen itself?

A stronger control (Phase 2f, planned before data): on 30,000 fresh documents of the same world, the model gave a confidence from 0 to 100 with each answer, and a second, independent answer was drawn. Accepting answers with confidence at least 82 (coverage 59.0%, matched to the checks) let 1,041 wrong results through (3.5×10^{-2} per document); accepting only confidence 100 let 0 through at 0.0% coverage. Accepting when two answers agree let 1,569 through (5.2×10^{-2}) at 93.5% coverage. The five checks on the same documents let 0 through (bound 1.0×10^{-4}) at 69.6% coverage. Both forms of self-screening were worse than the checks (one-sided Fisher $p < 10^{-100}$ for confidence and $p < 10^{-100}$ for agreement). The model never gave confidence 100, so accepting only that rule accepts nothing; its confident errors were mostly unresolved character-recognition misreads, many of which the checksum would have caught.

Clustering by vendor (exploratory, after the data)

The bound above treats documents as independent trials. Given the fixed set of 2,000 vendors they are: each document is drawn independently of the others, so the bound is exact for that vendor population. It does not carry over to a new set of vendors, because errors cluster by vendor. This is expected from the generator: the escapes L5 closes arise almost only among the 30% of vendors whose open orders share an amount. Four-check escapes came from 477 of the 2,000 vendors, a tenth of the vendors gave 56% of them, and the intraclass correlation was 0.014, a design effect of about 23 at 1,500 documents per vendor. The model's own errors cluster too (design effect about 17). With five-check escapes assumed to cluster like either, the effective sample shrinks to 130,000 to 180,000 documents and the bound for a new set of vendors becomes 1.7×10^{-5} to 2.3×10^{-5}. Taken one difficulty level at a time (750,000 documents each, zero escapes in each), the bound is 4.0×10^{-6} per level. A test meant to generalise over vendors needs many more vendors with fewer documents each: at 50,000 vendors and 60 documents per vendor the design effect would be about 1.9.

Run D: a world of 50,000 vendors (pre-registered)

We then ran that test. Only the world changed: a new seed and 50,000 vendors, built by the same rules (one to six open orders each, 30% of vendors with every order at one amount). Model, prompt, interface and the five checks were those of Run C, and the plan and analysis were committed before the first call. On documents of this world (about 60 per vendor, none seen before) the model was

wrong on 8.9%; four checks accepted 91.5% and let 11,754 errors through; five checks accepted 68.5% and let 0 through. The bound is 9.99×10^{-7} per output processed for this vendor population. The vendor-level design effect, taken as the larger of those of four-check escapes and of wrong outputs, was 1.78, an effective sample of 1.69 million documents, so the bound for a new set of vendors from the same generator is 1.78×10^{-6}. This met the pre-registered criterion of 10^{-5} and is close to, but not at, six nines; reaching 10^{-6} in this clustered sense would need about 5.4 million documents. Per difficulty level (750,000 documents each, zero escapes) the bound is 4.0×10^{-6}. The design effect is estimated from proxies, since there were no five-check escapes whose clustering could be measured. Run D used Claude Haiku 5.5 through the provider's interface, not the platform, and it is the only basis in this paper for any statement about vendors not seen before.

FIG. 4 · THE PRICE OF THE FIFTH CHECK

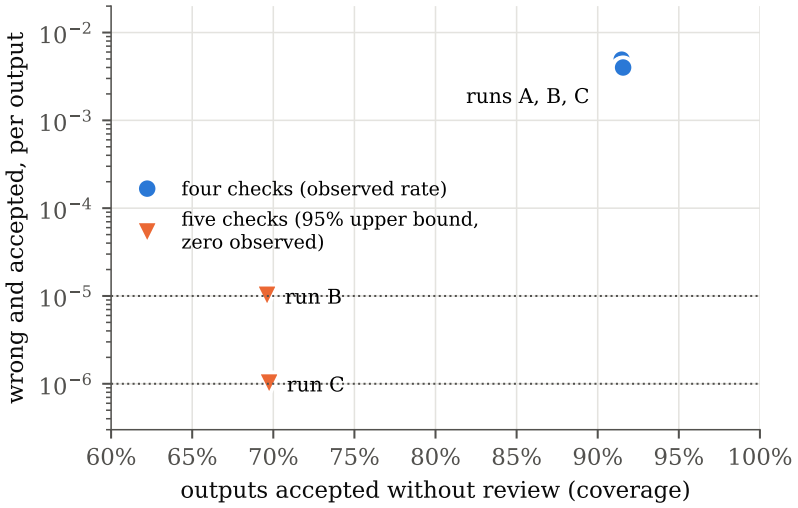

Figure 4. The price of the fifth check: coverage against the residual for four checks (observed rate) and five checks (95% upper bound, zero observed).

FIG. 5 · RUN C IN TEN BLOCKS

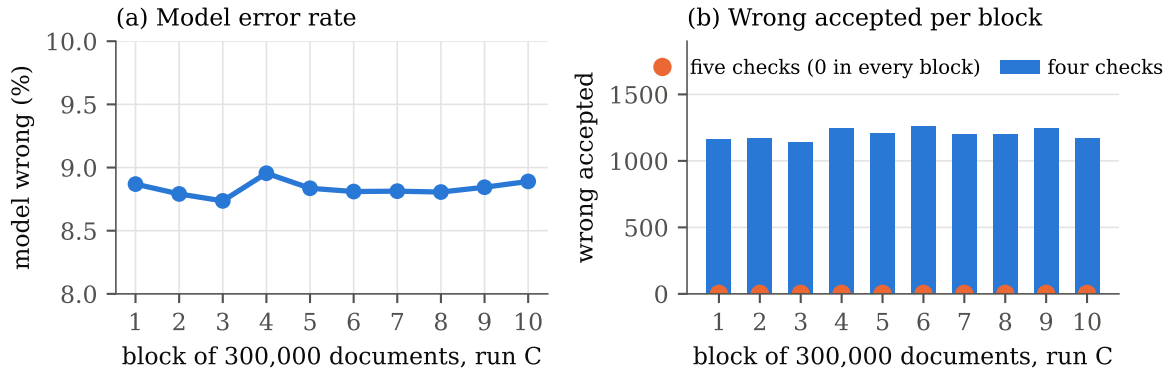

Figure 5. Run C in ten blocks of 300,000 documents. The model's error rate and the four-check escapes are stable; five checks let nothing through in any block.

The cost is automation

L5 sent 642,687 correct outputs to review in Run C. Automation fell from 91.6% to 69.7%. A rule that also used knowledge of which documents print a second invoice would keep 80.6%, but a deployed system does not have that knowledge, so we report it only as an upper reference.

An unexplained shift

The rate of the same-amount confusion was 4.2% in Run A and 3.4% in Run B, one day apart, with the same model name, prompt and document generator and an equal number of confusable documents. We do not know the cause. It is a small, concrete instance of drift, and a reason to treat any bound as belonging to the day and configuration on which it was measured.

8 Result 3: a direct count through a platform workflow

FIG. 6 · EXTRACTANDCHECK AS AN INTEGRATION WORKFLOW

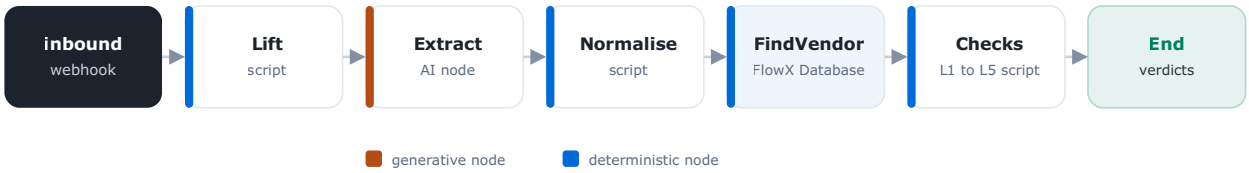

Figure 6. The same pipeline as a FlowX.AI integration workflow, ExtractAndCheck. The only generative node is Extract; the vendor registry and order ledger are a platform database holding the 2,000 vendors of the synthetic world.

The checks above were reference code. We rebuilt the pipeline as an integration workflow, ExtractAndCheck, on the FlowX.AI Platform: a model node, a script that normalises the record, a lookup in a platform database holding the 2,000 vendors and their orders, a script with the five layers, and an end node returning the verdicts (Figure 6). Documents were sent through an inbound webhook, one run per document. The whole test ran in a demo (non-production) environment. That environment offered no Claude model, so the model node used other providers' models with the same prompt and a five-field output schema.

Faithfulness of the port

For every run we compared the workflow's normalised record and all five layer verdicts with those of the reference implementation on the same model output. On 1,000 fresh documents with GPT-5.6 Terra in the model node all runs finished and there were 0 disagreements; the model was wrong on 83 (8.3%), 698 outputs were accepted under five checks and none of them was wrong. A first arm of 100 documents with GPT-4.1 Mini (wrong on 20, 57 accepted under five checks) also showed 0 disagreements. The model used from then on, GPT-6 Luna, was chosen after seeing these arms. With it, a first run of 300,000 fresh documents (plan amendment committed before the run) finished 299,639 runs with a result; 55 failed in the model node and were excluded, not retried or counted as correct. There were 0 disagreements; the model was wrong on 23,092 (7.7%); four checks let 61 of them through and five checks 0, a bound of 1.00×10^{-5} per output processed. That run sized the count below and is reported on its own, not pooled with it.

FIG. 7 · THE PLATFORM COUNT, BATCH BY BATCH

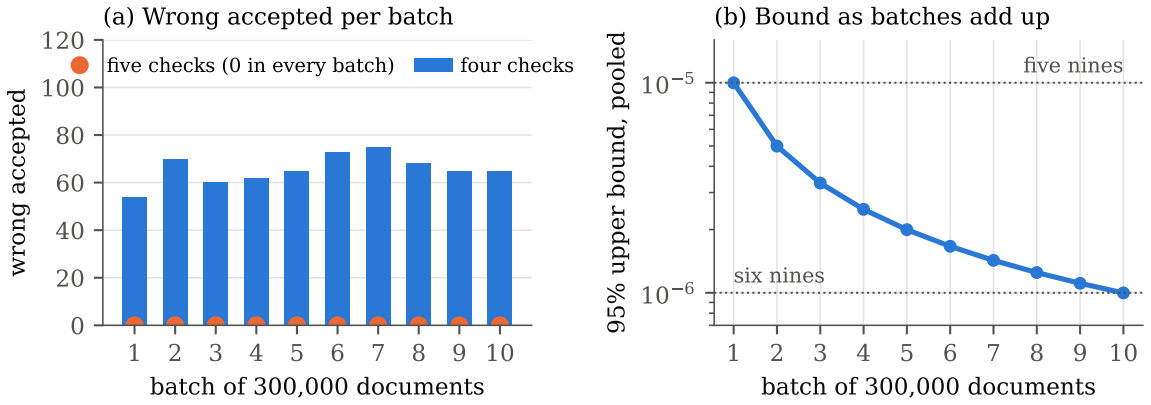

Figure 7. The platform count, ten batches of 300,000 documents. (a) Wrong outputs accepted by four checks and by five in each batch. (b) The pooled one-sided 95% upper bound on the residual with five checks as batches are added, reaching 9.99×10^{-7} at 2,999,805 finished documents.

The platform count

The plan, the pooling rule and the analysis were committed before the first document. Ten batches of 300,000 fresh documents of the Run C world, 3,000,000 in all and none sent before, went through ExtractAndCheck with GPT-6 Luna in the model node and the same prompt, schema, scripts, vendors and five checks. Each batch was sent at a fixed rate below the throughput the environment had held in a ceiling test. Every run was collected and compared with the reference implementation every five minutes, and sending would have stopped at the first wrong output accepted by the five checks. The measured quantity is the number of wrong accepted outputs among finished runs. Runs that did not finish are counted and reported; they were never retried and never counted as correct.

Table 3. Run C and the platform count, both on fresh documents of the same 2,000-vendor synthetic world.

	Run C	Platform count
Model	Claude Haiku 5.5	GPT-6 Luna
Where the checks ran	reference code	platform workflow
Documents with a result	3,000,000	2,999,805
Platform vs. reference disagreements	–	0
Model wrong, ε	8.8%	7.72%
Four checks: accepted	91.6%	92.30%
Four checks: wrong accepted	11,996	657
Five checks: accepted	69.7%	70.25%
Five checks: wrong accepted	0	0
Upper bound on r, 95%	9.99×10^{-7}	9.99×10^{-7}
Upper bound on r_{acc}, 95%	1.76×10^{-6}	1.42×10^{-6}

Of the 3,000,000 documents, 2,999,805 runs finished with a result, 42 started but did not finish, and 153 produced no collected run. On every finished run the platform's normalised record and five layer verdicts equalled the reference implementation's (0 disagreements). The model was wrong on 231,522 documents (7.72%). Four checks accepted 92.30% of outputs and let 657 wrong ones through; five checks accepted 70.25% and let 0 through (Table 3, Figure 7). The one-sided 95% upper bound on the residual is 9.99×10^{-7} per output processed, just below one in a million; among accepted outputs it is 1.42×10^{-6}. Results were consistent across parallel runs (Fisher exact tests on wrong rates, all $p > 0.1$), so they are pooled as the plan specified. The four checks let through a much smaller share of GPT-6 Luna's errors than of Claude Haiku 5.5's in Run C, so the fifth check had less to do here; we did not study why. With five checks both counts are zero.

What this shows is narrow. A second model, run end to end on the platform, behind the same five checks, gave no wrong accepted output in 2,999,805 fresh documents of the 2,000-vendor synthetic world, and the platform computed exactly the verdicts the reference code computes. The count inherits the vendor-clustering caveat of Result 2: it is a bound for those 2,000 vendors, not for new ones.

What is not accepted

The 29.75% of documents not accepted are not an error rate, and they need not all go to a person. The residual concerns only what is accepted automatically; what happens to the rest is a choice of the use case: reject it, retry the extraction, or send it to a person. Because no wrong output was accepted, they consist of every document the model got wrong (7.72%) and 22.03% of documents where the model was right but the system could not confirm it: all of the latter were refused by L5, because the vendor had more than one open invoice with that amount and the ledger could not tell which one was meant. The first part is the residual guarantee at work. The second is a property of the document

mix, and our mix was made hard on purpose (30% of vendors with every open order at one amount). On the 387 real invoices of Result 4, 84% of outputs were accepted. On real data the share not accepted has to be measured; it is not fixed by the method.

The platform count on new vendors (pre-registered)

To bound the platform's residual for vendors not seen before, we repeated the count on the world of Run D. Plan, criterion and analysis were committed before the first document: four batches of 150,000 fresh documents of the 50,000-vendor world (about 12 per vendor), the same workflow, model, prompt and five checks, and the criterion that no wrong output is accepted and the bound for new vendors is at most 10^{-5}. Of 600,000 documents, 599,952 runs finished and 48 did not; there were 0 disagreements with the reference implementation. The model was wrong on 46,836 (7.81%); four checks accepted 92.21% and let 120 errors through; five checks accepted 69.11% and let 0 through. The bound is 4.99×10^{-6} for this vendor population. With the vendor-level design effect taken, as in Run D, as the larger of those of four-check escapes and of wrong outputs (1.12), the effective sample is 536,694 documents and the bound for a new set of vendors from the same generator is 5.58×10^{-6}. The criterion was met. This is five nines for new vendors on the platform, not six; six would need about 3.3 million documents at this design effect. Of the 30.89% of documents not accepted, 7.81% were the model's errors and 23.08% correct outputs the system could not confirm.

Interruptions

Delivery was paused several times by infrastructure events, among them an expired key at the model provider, database restarts and a full message-broker disk. Each time, delivery resumed from the next unsent document. A guard stopped sending on any restart of a shared platform component; in one case a database pod was replaced rather than restarted, the guard did not catch it, and the batch continued through it with no failed run, after which the guard was extended to catch replacements. In the count on new vendors the extended guard stopped one batch when two pods were replaced by the cluster's autoscaler; the batch resumed from the next unsent document. Runs that failed because of the expired key failed before the model produced any output; those documents were sent again. A resent delivery sometimes produced a second run of a document, and in every case the earliest finished run is the one counted, so each document counts once. None of these events touches the scored outcome, which is decided by the model's output and the checks, and the run-by-run comparison with the reference implementation covers every counted run.

Throughput

To size the count we sent fresh documents at fixed rates with GPT-6 Luna in the model node (Figure 8). Every run in these checks that finished also agreed with the reference implementation. Throughput was limited first by a message topic with a single partition, so that only one of several model-worker instances received jobs, then by the number of concurrent jobs per instance, and finally by the number of model calls in flight against the provider's response time, which grew with load. These are properties of a demo environment with study settings, not of the product, and the study did not try to tune them further.

FIG. 8 · EARLY LOAD CHECKS

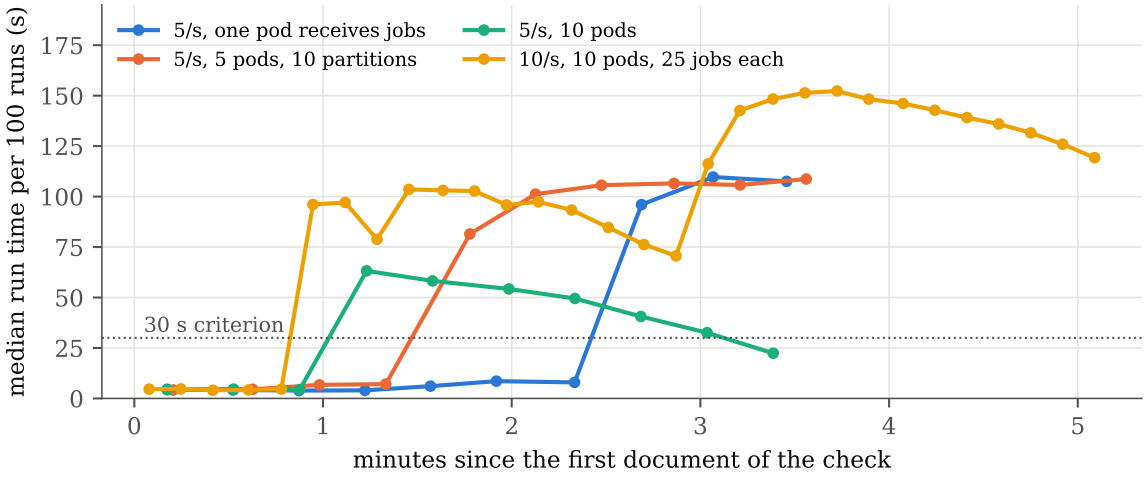

Figure 8. Median run time per 100 runs against time, for four early load checks in the demo environment. Each change (more partitions, more instances, more jobs per instance) delayed the backlog but none removed it at the rates tested.

An invented record

One incident during the build is worth recording. A wiring mistake sent the model an empty document; the model returned a complete, plausible payment record for a company that appears nowhere in the data, with an IBAN that passes the checksum. The registry check rejected it. Format and checksum checks do not protect against invention; checks against a system of record do.

9 Result 4: real invoices

Synthetic documents cannot show whether the model's real errors, and the checks' blind spot, look the same on real invoices. We ran the Phase 2 prompt and model unchanged on CIVQA, a public set of Czech and Slovak invoices with OCR text and human labels[24]. The plan, the frozen document list and the analysis were committed before any call. Eligible documents have an IBAN label that passes the checksum and all five fields labelled; the currency is derived from the text by a fixed rule. Registry and ledger are built from the labels of the *other* invoices of the same vendor, which is the main threat to validity here: such a registry is cleaner than a real one. Results are on the raw labels and provisional until two people adjudicate every disagreement.

On 387 invoices the model was wrong on 36 (9.3%, 95% CI 6.6% to 12.6%). Of those errors, 25 (69%, CI 52% to 84%) were of a kind no mutation operator produces, so the injected-error model fails here even more clearly than on synthetic documents. Whether checks miss real errors more often than injected ones could not be decided at this size (1 of 24 real errors missed against 0 of 116 injected, Fisher $p = 0.17$, on 116 invoices with at most 20 per vendor), though the direction matches Run A. The same-amount confusion was not observed among 62 invoices printing a second reference (bound 4.7%). With five checks 1 wrong output was accepted out of 387 (bound 1.2%), at 84% coverage; on inspection the label there holds the net amount while the model returned the total with VAT, which is also the label of another copy of the same invoice in the data. These numbers are percent-level and say nothing about nines; they say that the main finding of Result 1 is not an artefact of the generator.

10 What the results mean for a six-nines claim

- A residual error rate belongs to a configuration: model, prompt, checks and document mix. It is not a property of a platform or a model.
- On this synthetic task the claim that could be supported is: at most about one wrong accepted output per million processed, at 95% confidence, for a fixed set of vendors, with about 70% of outputs accepted without review. For vendors not seen before the supported figures are 1.78×10^{-6} (Run D, Claude Haiku 5.5 through the provider's interface), close to six nines but not at it, and 5.58×10^{-6} on the platform with GPT-6 Luna, five nines.
- The figure is per output processed, not the model's accuracy: the model was wrong on about 8% of documents. The checks do not correct those errors; they keep them from being accepted, together with correct outputs they cannot confirm, and the use case decides what happens to those: rejection, a retry or a person. How many of the latter there are depends on the documents.
- The result survived a change of model and a move into a platform workflow, on the same synthetic world, provided the port was verified run by run against the reference implementation.
- The composed shortcut would have certified a figure 5.3 times too low before the blind spot was found. Injected errors are useful for finding which checks are weak. They did not predict where the model fails.
- The remedy was not a better model or a second model as judge. It was a deterministic rule that refuses when the system of record cannot tell two answers apart, and another path for those cases.

11 Limitations

- **Synthetic documents.** About 7,000 orders rendered many times by one generator. Real documents, real character recognition and real ledgers will have error kinds this generator does not produce. The blind spot we found was one we could have predicted from the design; real ones may not be.
- **Two main models, one prompt, a few days.** Each bound is for its configuration. The laboratory runs used Claude Haiku 5.5 and the platform count used GPT-6 Luna; neither result says anything about the other model in the other setting. We observed an unexplained shift between two consecutive days.
- **Truth by construction.** This removes label noise, which is what makes a count of this size affordable. With real documents truth must come from outcomes such as settled payments or reconciliation, because human labels are not accurate enough to certify 10^{-6}.
- **The fifth layer was designed after seeing Run A.** It was then tested on fresh documents with a fixed criterion, but the task stayed the same. A new task needs its own search for blind spots.
- **Zero observed is not zero.** The bounds are upper limits at 95% confidence under exchangeability. They say nothing once the document mix changes.
- **Independence.** The Run C and platform bounds are exact for the fixed set of 2,000 vendors. Errors cluster by vendor, and for a new set of vendors the Run C bound becomes about 2×10^{-5} (Result 2). On the world of 50,000 vendors with few documents each, the new-vendor bounds are 1.78×10^{-6} (Run D, Claude Haiku 5.5 outside the platform) and 5.58×10^{-6} (GPT-6 Luna on the platform), both using a design effect estimated from proxies because no five-check escape was observed. Within-document clustering is known to break per-field guarantees in document extraction[5].
- **The platform count is not a production measurement.** It ran in a demo (non-production) environment, with study settings, on synthetic documents. It measures the residual of one configuration, not the platform's reliability, availability or behaviour under production load, and the infrastructure interruptions it met are not characterised here. The checks are study scripts running in the workflow, not product features. GPT-6 Luna was chosen after seeing two other models on small arms.
- **Simulation results** hold for the dependence structures simulated.

12 Conclusion

Wrapping a model step in deterministic checks can bring the residual error below one in a million on a task where a system of record can verify the output, and this can be demonstrated by counting. On our synthetic task it took a direct count of millions of documents, a fifth rule found only by direct counting, and the loss of about a fifth of the automation. The result held on a fresh world of 50,000 vendors, where it also gave a bound of 1.78×10^{-6} for vendors not seen before, and it held when the same checks ran as a workflow on the FlowX.AI Platform with a different model: no wrong output was accepted in 2,999,805 documents run end to end in a demo environment, and the platform's verdicts matched the reference code on every run. Repeated on the platform with 50,000 vendors, it gave 5.58×10^{-6} for vendors not seen before. None of this is a statement about real invoices or production systems. The shortcut of composing an error rate with a miss rate from injected errors did not survive contact with a real model, because the model fails where the checks are blind. For anyone stating such a figure, the practical consequence is to name the configuration, state the coverage next to the residual, and count.

Data and code

The document generator, checks, mutation operators, run plans, analysis scripts, decision log and claims ledger are in the project repository; every number in this paper is generated from the result files by a script.

References

[1] *Axiomat: Agents that Cannot Hallucinate.* Launch YC, https://www.ycombinator.com/launches/X3M-axiomat-agents-that-cannot-hallucinate. Accessed 2026-11-12. Vendor statement.

[2] R. W. Butler and G. B. Finelli. *The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software.* IEEE Transactions on Software Engineering, 19(1):3-12, 1993.

[3] C. J. Clopper and E. S. Pearson. *The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial.* Biometrika, 26(4):404-413, 1934.

[4] D. E. Eckhardt and L. D. Lee. *A Theoretical Basis for the Analysis of Multiversion Software Subject to Coincident Errors.* IEEE Transactions on Software Engineering, SE-11(12), 1985.

[5] B. Gurram. *Valid Per-Field Selective Risk Control for Document Extraction: Three Failure Modes, a Validity Ladder, and When Conditioning Pays.* arXiv:2608.14639, 2026.

[6] J. A. Hanley and A. Lippman-Hand. *If Nothing Goes Wrong, Is Everything All Right? Interpreting Zero Numerators.* JAMA, 249(13):1743-1745, 1983.

[7] I. Ivrissimtzis, M. Houliston, S. Concannon and G. Roberts. *Zero-Failure Testing of Binary Classifiers.* arXiv:2407.03979, 2024.

[8] Y. Jia and M. Harman. *An Analysis and Survey of the Development of Mutation Testing.* IEEE Transactions on Software Engineering, 37(5):649-678, 2011.

[9] Y. Jin, Y. Wang, Y. Han, C. Y. S. Kwan, W. Song, J.-C. K. Chiu, C. H. Wong, T. K. Wong, K. Zhu Ke and J. Zhao. *Perception, Layout, and Validation: Calibrated Confidence for Reliable Straight-Through Processing of Financial Documents.* arXiv:2609.20110, 2026.

[10] R. Just, D. Jalali, L. Inozemtseva, M. D. Ernst, R. Holmes and G. Fraser. *Are Mutants a Valid Substitute for Real Faults in Software Testing?* Proceedings of FSE 2014, pp. 654-665, 2014.

[11] E. Kim, A. Garg, K. Peng and N. Garg. *Correlated Errors in Large Language Models.* Proceedings of the 42nd International Conference on Machine Learning, PMLR 267, pp. 30038-30066, 2025.

[12] J. C. Knight and P. E. Ammann. *An Experimental Evaluation of Error Seeding as a Program Validation Technique.* Proceedings of the 10th Annual Software Engineering Workshop, NASA Goddard Space Flight Center, 1985.

[13] J. C. Knight and N. G. Leveson. *An Experimental Evaluation of the Assumption of Independence in Multiversion Programming.* IEEE Transactions on Software Engineering, 12(1):96-109, 1986.

[14] V. Kotte. *When Can Conformal Risk Control Certify LLM Outputs? Bounds, Impossibility, and Adaptation for Structured Generation.* arXiv:2606.29054, 2026.

[15] Z. Li. *Post-Hoc Trajectory-Risk Certification for Modular LLM-Based Security Agents.* arXiv:2608.05199, 2026.

[16] B. Littlewood and L. Strigini. *Validation of Ultrahigh Dependability for Software-Based Systems.* Communications of the ACM, 36(11):69-80, 1993.

[17] V. Magesh, F. Surani, M. Dahl, M. Suzgun, C. D. Manning and D. E. Ho. *Hallucination-Free? Assessing the Reliability of Leading AI Legal Research Tools.* arXiv:2405.20362, 2024.

[18] S. Messing. *Hidden Measurement Error in LLM Pipelines Distorts Annotation, Evaluation, and Benchmarking.* Advances in Neural Information Processing Systems, 2026. arXiv:2604.11581.

[19] E. Meyerson, G. Paolo, R. Dailey, H. Shahrzad, O. Francon, C. F. Hayes, X. Qiu, B. Hodjat and R. Miikkulainen. *Solving a Million-Step LLM Task with Zero Errors.* arXiv:2511.09030, 2025.

[20] R. Natella, D. Cotroneo, J. A. Durães and H. S. Madeira. *On Fault Representativeness of Software Fault Injection.* IEEE Transactions on Software Engineering, 39(1):80-96, 2013.

[21] B. Raduta, H. Velicu, A. Preda and S. Chiricescu. *Zero Hallucination, by Construction: Hallucination-Aware Layered Oversight for Trustworthy Enterprise AI.* arXiv:2607.17883, 2026.

[22] J. Ron, B. Baudry and M. Monperrus. *N-Version Programming with Coding Agents.* arXiv:2606.20158, 2026.

[23] B. Răduță, H. Velicu, A. Preda and Ș. Chiricescu. *The Graph Is Deterministic, Even When the Nodes Are Not: Reliable Agent Execution via Layered State-Machines.* Technical paper, Version 1.1, FlowX.AI, August 2026.

[24] Š. Ščavnická, M. Štefánik and P. Sojka. *Document Visual Question Answering with CIVQA: Czech Invoice Visual Question Answering Dataset.* Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2023, pp. 23-34, Tribun EU, 2023. https://www.fi.muni.cz/usr/sojka/papers/scavnicka-stefanik-sojka-raslan-2023.pdf

[25] D. X. Wu, S. Kapur, A. Sahai and S. Russell. *Synthetic Error Injection Fails to Elicit Self-Correction in Language Models.* arXiv:2512.02389, 2025.

FlowX.AI

FlowX.AI builds and orchestrates enterprise-grade AI agents for regulated industries, with a focus on banking, financial services, and insurance. This paper measures the residual error left after deterministic checks on one model step, including a direct count with the same checks running as an integration workflow on the FlowX.AI Platform in a demo (non-production) environment.

FlowX.AI Technical Paper · SEXTANT v1.0 · November 2026 · Bogdan Răduță, Head of Research · bogdan.raduta@flowx.ai · DOI: 10.5281/zenodo.24339368